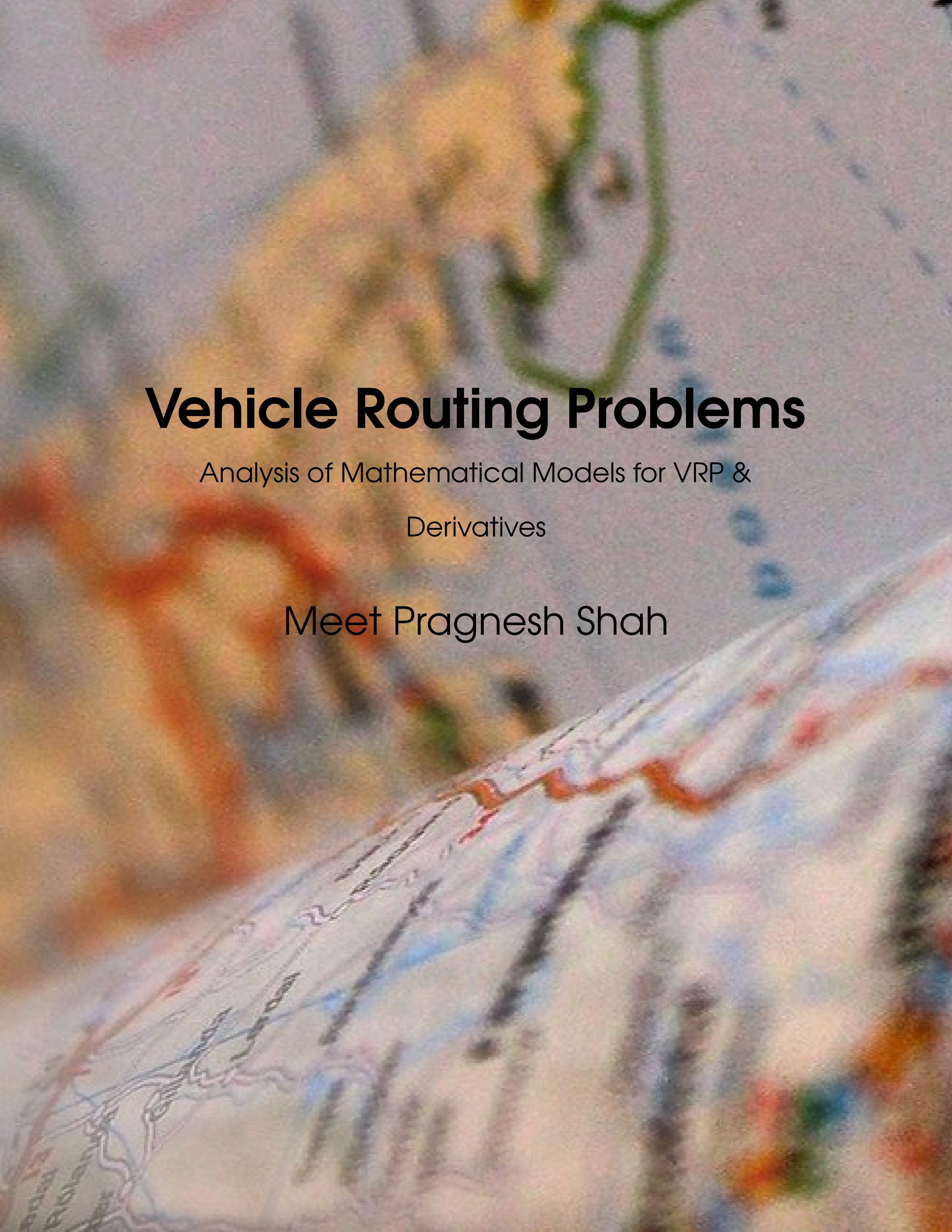


Vehicle Routing Problems

Analysis of Mathematical Models for VRP & Derivatives

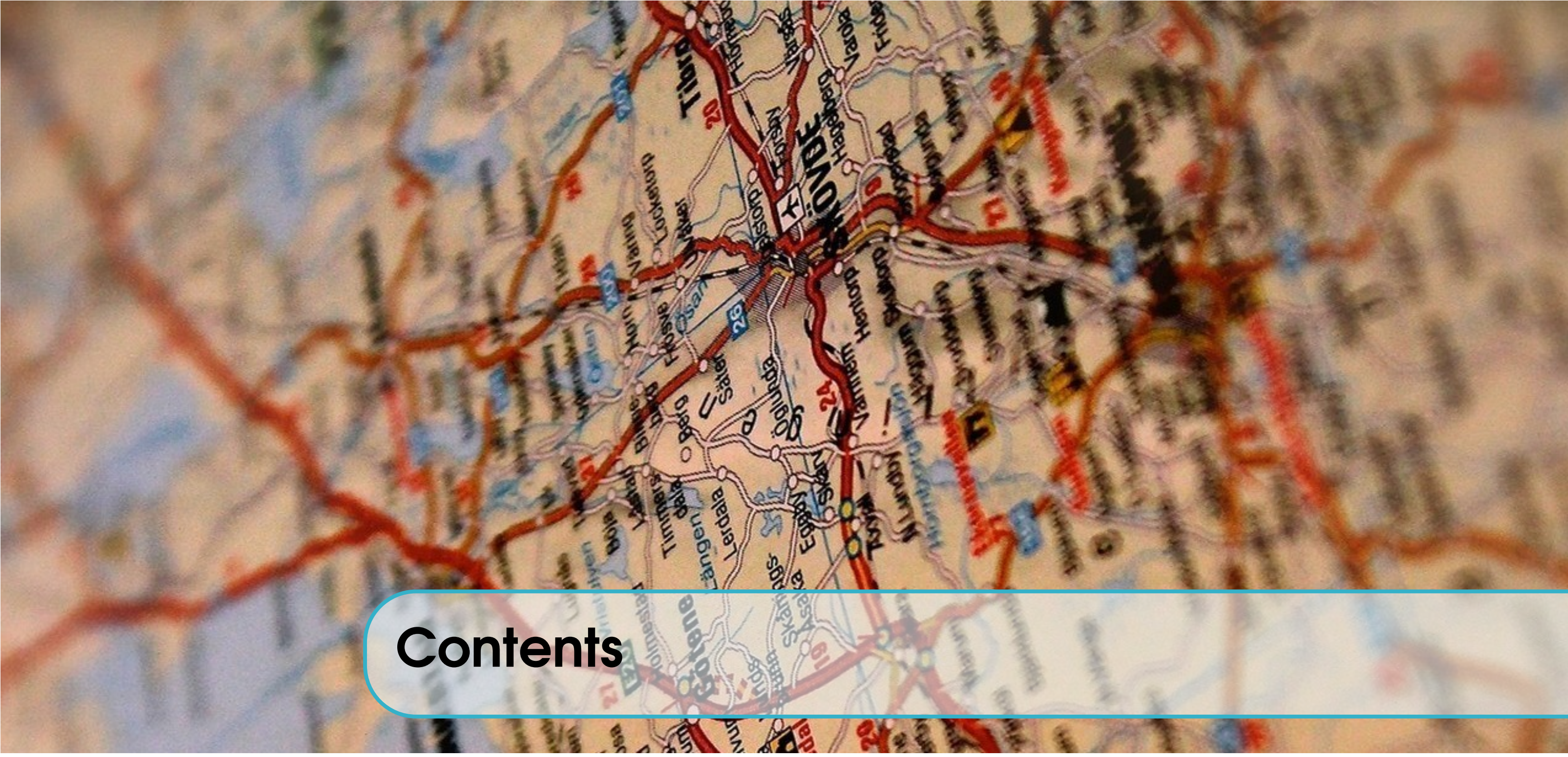
Meet Pragnesh Shah

WINTER RESEARCH INTERNSHIP

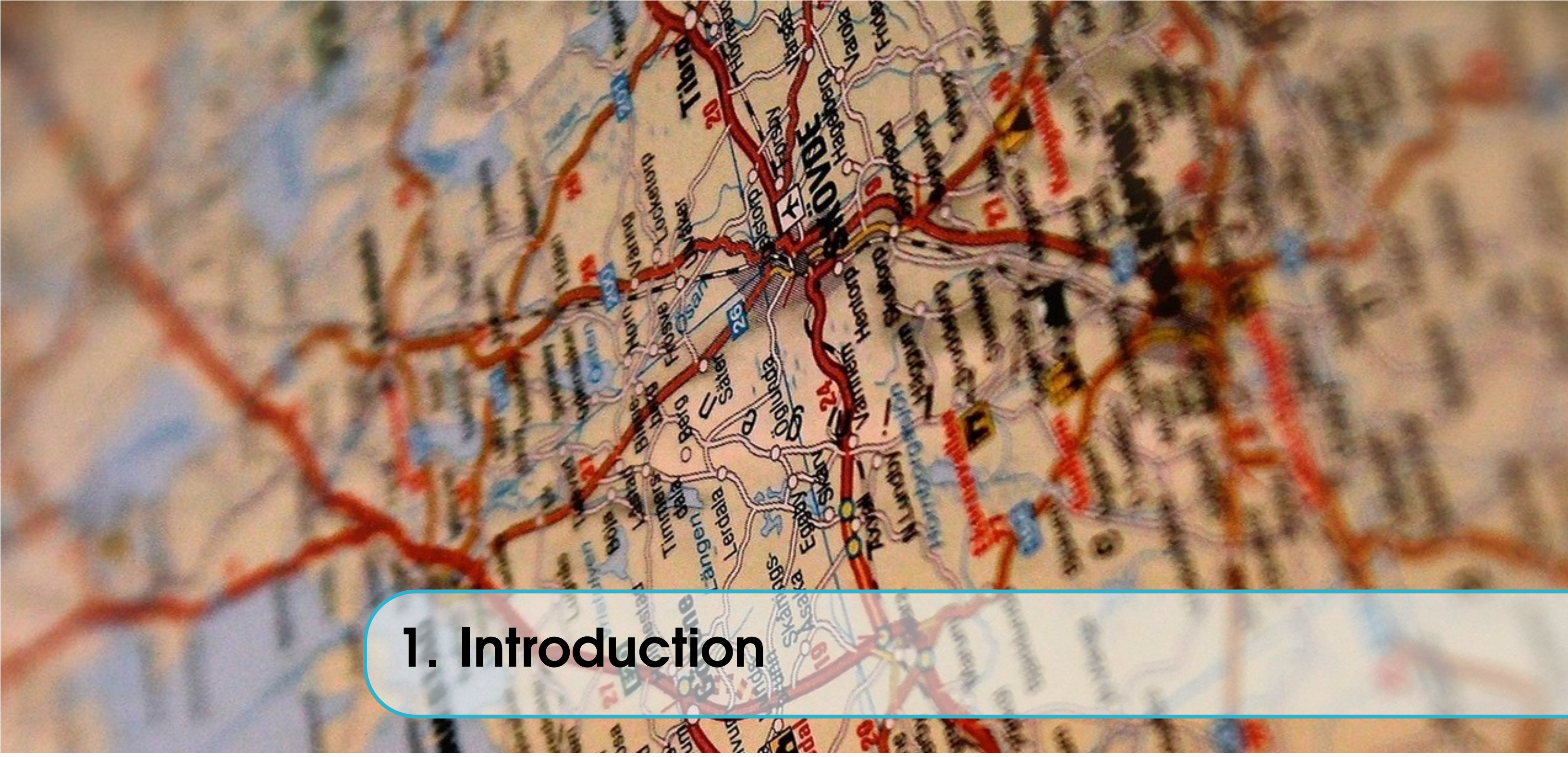
INDIAN INSTITUTE OF MANAGEMENT - AHMEDABAD

github.com/meetshah1995/IIMA-Intern

This research was done under the supervision of Prof. Sachin Jayaswal within a total of 5 weeks, from November 26th of 2014 to January 1st of 2015



Contents



1. Introduction

1.1 Introduction

The entire research internship duration was quite a walkthrough a lot of detailed MatheMatical Modelling of complex yet real-life optimization problems. Running an industry can be done even by a layman , but what takes you a mile ahead is how you use available resources judiciously and optimally and come up with ingenious methods of getting work done. One of many effects this research internship had on my mindset is that I got myself introduced to the genre of Operations Research both in theory and practical aspects.

> This internship will have a long and lasting effect on the decisions I take in the future. I am completely thankful to Prof. Sachin Jayaswal to consider me worthy of the position and introduce me to this completely new genre in research : IEOR.

1.2 Abstract

During my short stay at IIM , I touched the topic of Vehicle Routing and Travelling Salesman problems, then went into the intricacies of the mathematical models to describe these problems , their interpretations and analysis of different models and related complexities.

This effort at research was a problem-based learning where I was supplied with several problems of varied genres under VRP to solve and hence get the chance to go through some very good papers on those topics varying from k-periodic symetric travelling salesman problems to vehicle routing problem with time windows.The allocation of problems in increasing order of Difficulty was extremely critical to the learning process alongside some good problem removal sessions with Sir.

I perused as much as 13 papers during my stay here and got an insight into the intricacies of Vehivle Routing Problem. I have given a brief literature review of each paper in the fore-coming sections which describes what made the paper unique and how it contributed to my learning process.

1.3 Software & Packages Used

- Microsoft Excel
  - OpenSolver
  - C-Solver
- IBM CPLEX
- AMPL

Each of these above tools for implementing and solving mathematical models is perfect in their own fields and I learnt that the choice of the tool to be used highly depends upon the problem statement and the format of the question.

1.4 References

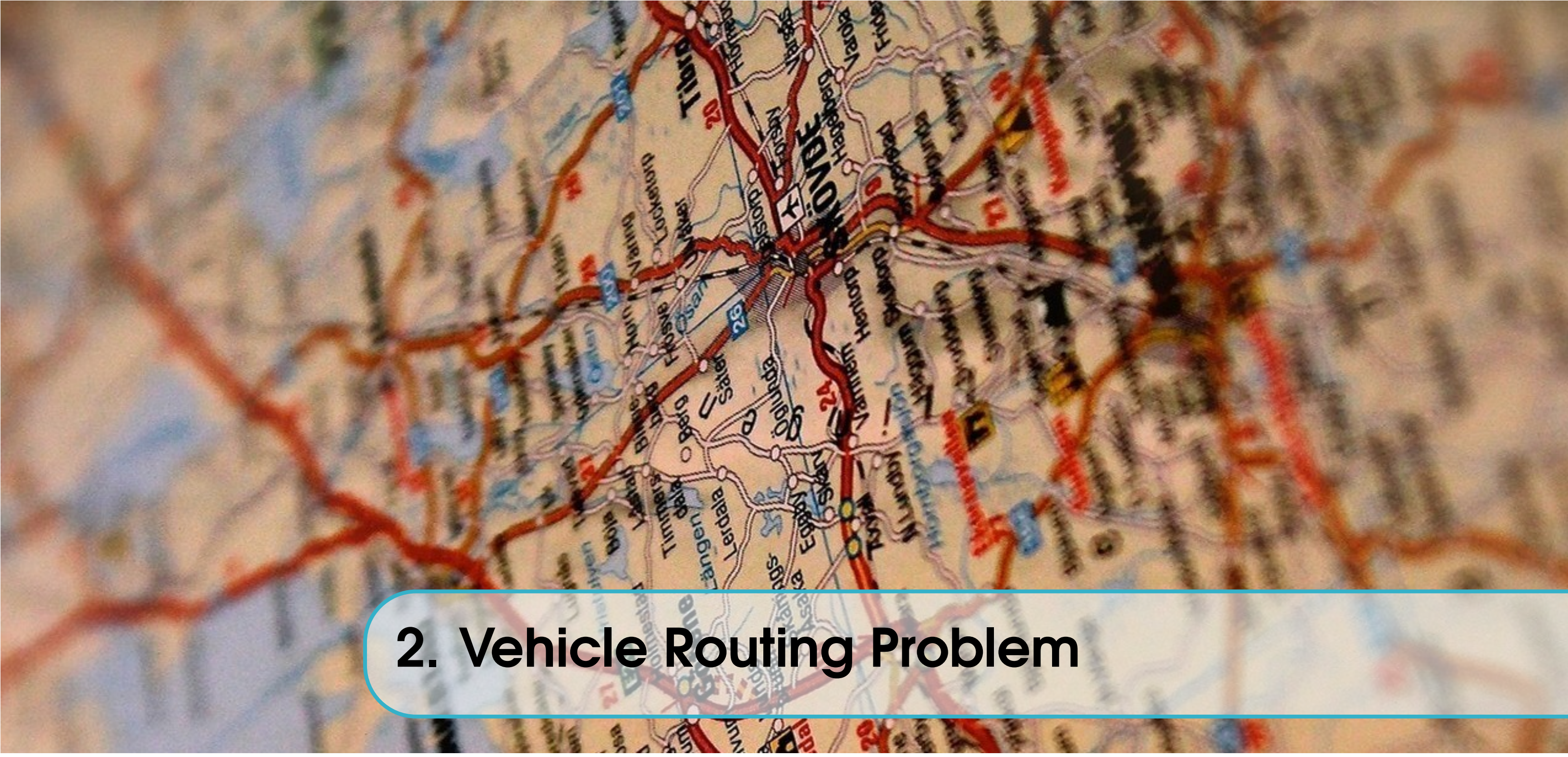
Since I was entirely new to this genre of research , I had to go through a couple of textbooks and popular papers on Vehicle Routing Problem and its derivatives. I have listed down a few popular papers below which I will be reviewing in this report.

- The two-period TSP applied to Milk-Collection in Ireland
- Integer Programming formulations of multiple salesman problems
- The Vehicle Routing Problem: An overview of exact algorithms
- A Survey on the Vehicle Routing Problem and Its Variants
- A Modelling and Optimization Framework for Real-World Vehicle Routing Problems

1.5 TextBooks

Introduction to Operations Research, *Hiltier*| *Lieberman* | *Nag* | *Basu*, Third Edition, 2010.

Model Building in Mathematical Programming, *Williams*, Fifth Edition, 2010.



2. Vehicle Routing Problem

2.1 First ideas

The vehicle routing problem (VRP) is a combinatorial optimization and integer programming problem seeking to service a number of customers with a fleet of vehicles. Proposed by Dantzig and Ramser in 1959, VRP is an important problem in the fields of transportation, distribution and logistics. Often the context is that of delivering goods located at a central depot to customers who have placed orders for such goods. The objective of the VRP is to minimize the total route cost.

Determining the optimal solution is an NP-complete problem in combinatorial optimization, so in practice heuristic and deterministic methods have been developed that find acceptably good solutions for the VRP. A non-technical explanation exists for why the VRP is much more challenging than might be expected.

2.2 Derivatives of Vehicle Routing Problem

Several variations and specializations of the vehicle routing problem exist:

- Travelling Salesman Problem (TSP):
  A TSP is essentially a VRP with a single depot and a single vehicle, in other words the TSP can be seen as an over-simplified version of the VRP and is perfect to start as a beginner to VRPs.
- Vehicle Routing Problem with Pickup and Delivery (VRPPD):
  A number of goods need to be moved from certain pickup locations to other delivery locations. The goal is to find optimal routes for a fleet of vehicles to visit the pickup and drop-off locations.
- Vehicle Routing Problem with LIFO:
  Similar to the VRPPD, except an additional restriction is placed on the loading of the vehicles: at any delivery location, the item being delivered must be the item most recently picked up. This scheme reduces the loading and unloading times at delivery locations because there is no need to temporarily unload items other than the ones that should be dropped off.

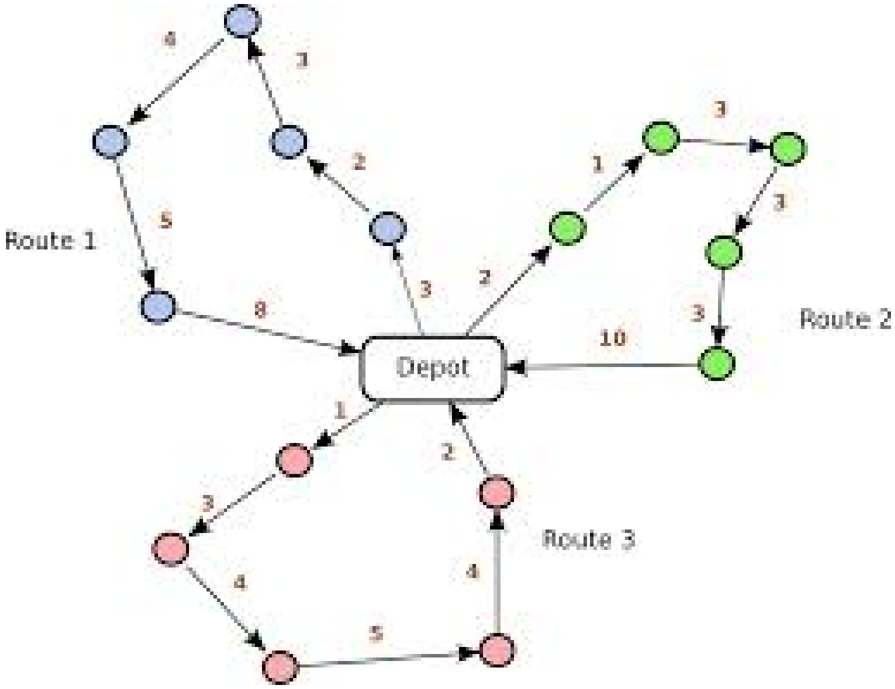


Figure 2.1: Picuture of Single Depot n-Vehicle Routing , n=3.

- Vehicle Routing Problem with Time Windows (VRPTW):
  The delivery locations have time windows within which the deliveries (or visits) must be made. In computational complexity theory, this problem is known to be NP-hard.
- Capacitated Vehicle Routing Problem (with or without Time Windows): CVRP or CVRPTW. The vehicles have limited carrying capacity of the goods that must be delivered. Vehicle Routing Problem with Multiple Trips (VRPMT): The vehicles can do more than one route.
- Open Vehicle Routing Problem (OVRP):
  Vehicles are not required to return to the depot. Several software vendors have built software products to solve the various VRP problems. Numerous articles are available for more detail on their research and results.

2.3 Indutrial Application of VRP and its derivatives

The Application of the the VRP and its derivatives is boundless in today's era of door to door delivery. The very presence of better trasnport connectivity have boosted long distance business and hence burdened the transport section. With nations fighting over fossil fuel, it is of utmost priority to the transport section to use the fuels judiciously and engage lowest cost alongside maintaining customer satisfaction. A few day-to-day experiences where Vehicle Routing can save millions of rupees are :

- Bread Distibution
- Online Market Supply Chain
- Industrial Transport
- Grains and vegetables Transport

2.4 Mathematical Model of Simple Vehicle Routing Problem

2.4.1 Model Definition

In this section, the notation and formulation of the VRP are introduced. Let $C = [1, 2, ..., n]$ be the set of customers, each customer i with a positive integer demand di, and let $V = [0] \cup C$, where 0 is the depot (with no demand). Each $arc(i, j) \quad \forall i, j \quad \in V$ has a nonnegative travel cost c_{ij} (usually distances). These costs are assumed to satisfy the triangle inequality. The VRP is defined on a directed graph $G = (V, A)$, where A is the set of $arcs[(i, j) \quad /i, j \in V, \quad ihspace.3cm j = 6]$. A set of K identical vehicles, each with capacity $Q \in Z^+$, is available at the depot.We assume that K is equal to K_{min}, the minimum number of vehicles needed to serve all the customers, which is determined as $K_{min} = \frac{d(C)}{Q_e}$, where d(C) is the sum of all the demands.

The objective function is the total travel cost and can be expressed as :

$$\sum_{(i,j)\in A} \sum_{k=1}^{K} c_{ij} x_{ijk}$$

where x_{ij} is a binary variable that takes value one if vehicle k travels directly from i to j (and zero otherwise). Defining variables $y_{ik} \geq 0$ as the demand of customer $i \in C$ delivered by vehicle $k = 1, 2, ..., K$, the vehicle flow formulation for the VRP :

$$\min \sum_{(i,j)\in A} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{2.1}$$

$$s.t. \sum_{i\in I} \sum_{k=1}^{K} x_{ijk} \geq 1 \qquad \forall j \in C \tag{2.2}$$

$$\sum_{j\in I} \sum_{k=1}^{K} x_{0jk} = K \tag{2.3}$$

$$\sum_{i\in I} x_{ipk} - \sum_{j\in I} x_{pjk} = 0 \qquad p \in I, k = 1, 2, 3, 4....K \tag{2.4}$$

$$\sum_{i\in S} \sum_{j\in S} x_{ijk} \leq |S| - 1 \qquad S \subseteq C, k = 1, 2, 3, 4....K \tag{2.5}$$

$$y_{ik} \leq d_i \sum_{j\in I} x_{ijk} \qquad i \in C, k = 1, 2, 3, 4....K \tag{2.6}$$

$$\sum_{i\in C} y_{ik} \leq Q \qquad k = 1, 2, 3, 4....K \tag{2.7}$$

$$\sum_{k=1}^{K} y_{ik} = d_i \qquad i \in C \tag{2.8}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j) \in A, k = 1, 2, 3, 4....K \tag{2.9}$$

$$y_{ik} \geq 0 \qquad i \in C, k = 1, 2, 3, 4....K \tag{2.10}$$

2.4.2 Constraints and their Interpretation

- (2.1) state that every customer must be visited by at least one vehicle
- (2.2) impose that exactly K routes are performed.
- (2.3) indicate that if vehicle k visits node p, then it must leave it (flow conservation constraints)
- (2.4) are the classical subtour elimination constraints.
- (2.5) guarantee that the customer can only be served if the vehicle visits it.
- (2.6) limit to Q the maximum load of each vehicle while
- (2.7) ensure that the entire demand of each node is satisfied.

2.4.3 Implementation in IBM CPLEX

```
/**********************************************************************
 *Vehicle Routing Problem
 *6 employees, 1 Office, minimum 2 tours for 3-seater cars.
 *Author : meetshah1995
 * Min Sum[i,j](Cij*Xijk) s.t.
 *              Sum[i,i!=j](Xijk)=1
 *              Sum[j,i!=j](Xijk)=1
 *              Sum[k,i!=j](Xijk)=1
 *              Sum[i]Sum[k](Xi7k)=2
 *              Sum[j]Sum[k](X7jk)=2
 *              0<=Xij<=1
 *********************************************************************/
#include<stdio.h>
#include<conio.h>
#include<iostream>
#include<fstream>
#include<iosfwd>
#include<string>
#include <deque>
#include <sstream>
#include <time.h>
#include <stdlib.h>
#include <vector>//for vectors
#include <math.h>
#include <cplex.h>
#include <ilcplex/ilocplex.h>
#include <ilconcert/ilosys.h>

using namespace std;

ILOSTLBEGIN

int main(int argc, char **argv)
{
        IloEnv env;
    try
        {
                IloArray<IloNumArray> distance(env);

                //Model Initialization
                IloModel model(env);
                //Taking Data from Distance Data File
                ifstream in;
                in.open("carpool-data.txt",ios::in);
                in >> distance;
                in.close();
                cout<<distance<<endl;

                //2D Array for Results
                IloArray<IloNumArray> X_val(env, 7);
                for (int i = 0; i < 7; i++)
                {
                        X_val[i] = IloNumArray(env,7);
                }

                IloArray<IloNumArray> Y_val(env, 7);
                for (int i = 0; i < 7; i++)
                {
                        Y_val[i] = IloNumArray(env,7);
                }

                //2D Array for Xij
                IloArray<IloNumVarArray> X(env, 7);
                for (int i = 0; i < 7; i++)
                {
                        X[i] = IloNumVarArray(env,7,0,1,ILOINT);
                }

                //2D Array for Yij
                IloArray<IloNumVarArray> Y(env, 7);
                for (int i = 0; i < 7; i++)
                {
                        Y[i] = IloNumVarArray(env,7,0,1,ILOINT);
                }


                // Defining Obj to be equal to sum-product(Dij*Xij)
                IloExpr Obj(env);
                for (int i = 0; i < 7; ++i)
                {
                        for (int j = 0; j < 7; j++)
                        {
                                Obj += distance[i][j]*X[i][j];
                        }
                }

                for (int i = 0; i < 7; ++i)
                {
                        for (int j = 0; j < 7; j++)
                        {
                                Obj += distance[i][j]*Y[i][j];
                        }
                }

                IloObjective Objective = IloMinimize(env);
                model.add(Objective);
                Objective.setExpr(Obj);
                Obj.end();

                //Constraints for the model

                //Ensuring that one edge is only on one tour
                IloExpr Exclusion(env);
                for (int i = 0; i < 7; i++)
                {
                        for (int j = 0; j < 7; j++)
                        {
                                Exclusion += X[i][j] + Y[i][j];
                                model.add(Exclusion == 1);
                        }
                }
                Exclusion.close();

                //outgoing
                for (int i = 0; i < 7; i++)
                {
                        IloExpr Sum(env);
                        for (int j = 0; j < 7; j++)
                        {
                                if(i != j)
                                        Sum += X[i][j];
                        }
                        model.add(Sum == 1);
                        Sum.end();
                }

                for (int i = 0; i < 7; i++)
                {
                        IloExpr Sum(env);
                        for (int j = 0; j < 7; j++)
                        {
                                if(i != j)
                                        Sum += Y[i][j];
                        }
                        model.add(Sum == 1);
                        Sum.end();
                }

                //incoming
                for (int j = 0; j < 7; j++)
                {
                        model.add(X[j][j]==0);
                        IloExpr Sum(env);
                        for (int i = 0; i < 7; i++)
                        {
                                if(i != j)
                                        Sum += X[i][j];
                        }
                        model.add(Sum == 1);
                        Sum.end();
                }
```

```
for (int j = 0; j < 7; j++)
{
        model.add(Y[j][j]==0);
        IloExpr Sum(env);
        for (int i = 0; i < 7; i++)
        {
                if(i != j)
                        Sum += Y[i][j];
        }
        model.add(Sum == 1);
        Sum.end();
}

//Ensuring 2 entries and exits for the Office node
IloExpr Double(env);
for (int i = 0; i < 7; i++)
{
        Double += X[6][i];
        Double += X[i][6];
}
for (int i = 0; i < 7; i++)
{
        Double += Y[6][i];
        Double += Y[i][6];
}
model.add(Double == 2);
Double.end();

//SECs
/*for (int i = 1; i < 7; i++)
{
        for (int j = 1; j < 7; j++)
                if(i!=j)
        {
                IloExpr sec(env);
                sec = U[i] - U[j] +7*X[i][j];
                model.add(sec <= 6);
                sec.end();
        }
}*/

// Optimize
IloCplex cplex(model);
cplex.setOut(env.getNullStream()); // if we get an empty
    stream in return
cplex.solve();//solving the MODEL
if (cplex.getStatus() == IloAlgorithm::Infeasible) // if
    the problem is infeasible
{
        env.out() << "Problem Infeasible" << endl;
}
for(int i = 0; i < 7 ; i++)
{
        cplex.getValues(X_val[i], X[i]);
}
```

```
                cout<<X_val<<endl;
                // Print results
                cout<< "Objective Value = " << cplex.getObjValue() <<
                    endl;
                cout<<"X = "<<X_val<<endl;
        }
        catch(IloException &e)
        {
                env.out() << "ERROR: " << e << endl;
        }
        catch(...)
        {
                env.out() << "Unknown exception" << endl;
        }
        env.end();
        return 0;
}
```

3. Paper Reviews

3.1 The two-period TSP applied to Milk-Collection in Ireland

Author(s) : Martin Butler, Paul Williams & Leslie-Ann Yarrow

In this classical paper , the author gives a mild introduction to k-period symmetric TSPs and then builds on them to apply the same model on Milk Collection in Ireland.

The paper essentially is a problem solving technique for 2 period symmetric TSP which needs an optimal route for milk collection where there are a few *everyday farms* which have to be visited daily and a few *everyday other farms* which have to be visited only once in two days. Using a clever implementation of the famous MTZ subtour elimination constraints , this problem is essentially shortened to a Multiple Traveling Salesman Problem.

3.2 Integer Programming formulations of multiple salesman problems

Author(s) : Imdat Kara & Tolga Bektas

In this paper, the authors extend the classical multiple traveling salesman problem (mTSP) by imposing a minimal number of nodes that a traveler must visit as a side condition. Single and multidepot cases and propose integer linear programming formulations for both, with new bounding and subtour elimination constraints. It is shown that several variations of the multiple salesman problem can be modeled in a similar manner. Computational analysis shows that the solution of the multidepot mTSP with the proposed formulation is significantly superior to previous approaches.

This paper is a example of power of *Kulkarni & Bhave* subtour eliminiation constraints. This example shows the excellence of *Kulkarni & Bhave* SECs over MTZ SECs in standard TSPs both in number constraints and space complexity.

3.3 The Vehicle Routing Problem: An overview of exact algorithms

Author(s) : Gilbert Laporte

In this classical must-read TSP paper , the author gives an extremely detailed introduction to solution methods to symmetric TSPs, which generally form the .

The paper essentially is a problem solving technique for 2 period symmetric TSP which needs an optimal route for milk collection where there are a few *everyday farms* which have to be visited daily and a few *everyday other farms* which have to be visited only once in two days. Using a clever implementation of the famous MTZ subtour elimination constraints , this problem is essentially shortened to a Multiple Travelling Salesman Problem.

3.4 A Survey on the Vehicle Routing Problem and Its Variants

Author(s) : Suresh Nanda Kumar & Ramasamy Panneerselvam

This paper exhaustively enlists all the possible variants of the VRP, which find application in the industrial logistics optimization.

This paper conducts a literature review on the recent developments and publications involving the vehicle routing problem and its variants, namely vehicle routing problem with time windows (VRPTW) and the capacitated vehicle routing problem (CVRP) and also their variants. The VRP is classified as an NP-hard problem. Hence, the use of exact optimization methods may be difficult to solve these problems in acceptable CPU times, when the problem involves real-world data sets that are very large. The vehicle routing problem comes under combinatorial problem. Hence, to get solutions in determining routes which are realistic and very close to the optimal solution, examples of using heuristics and meta-heuristics are given. In this paper we discuss the various exact methods and the k-opt heuristics and meta-heuristics used to solve the VRP and its variants.

3.5 A new approach to solving the multiple traveling salesperson problem using genetic algorithms

Author(s) : Arthur E. Carter & Cliff T. Ragsdale

In this revolutionary and intellectual stimulating paper , the authors gives a mild introduction to well known Genetic Algorithms and cleverly applies a them to solve the multiple travelling salesman problem with it using lower computational power and producing similar results.

.

This paper focuses on solving the MTSP with genetic algorithms (GAs) using standard TSP chromosomes and operators. This paper proposes a new GA chromosome and related operators for the MTSP and compares the theoretical properties and computational performance of the proposed technique to previous work. Computational testing shows the new approach results in a smaller search space and, in many cases, produces better solutions than previous techniques. This technique can serve as a basic solver for huge databases as the space complexity of the algorithm is almost neglible as to standard Mathematical Model solving using MTZ constraints as described in 3.2 .

Cheers, Meet Shah